UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Formação de Equipes e Gerência de Projetos Ágeis : Comparando sua Aplicação na Academia e na Indústria

Trabalho de Conclusão de Curso

Lucas Oliveira De Marchi

São Cristóvão – Sergipe

2017

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Lucas Oliveira De Marchi

Formação de Equipes e Gerência de Projetos Ágeis : Comparando sua Aplicação na Academia e na Indústria

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informaçãp.

Orientador(a): Prof. Dr. Rogério P.C. do Nascimento
Coorientador(a): Prof. MBA. PMP. Alércio Bressano Dória Mendonça

São Cristóvão – Sergipe

2017

Este trabalho é dedicado aos jovens líderes que dedicam seus dias para levar a sociedade à realização de grandes feitos.

Agradecimentos

Agradeço à Deus, à família, aos amigos, ao orientador e professores do DCOMP, por seu apoio incondicional durante o curso.

A ACONE pela oportunidade de me dedicar à melhoria de seus procesos.

A GoLife Company pela consultoria e auxílio na formação deste jovem profissional em gerenciamento de projetos.

"Far better is it to dare mighty things, to win glorious triumphs, even though checkered by failure... than to rank with those poor spirits who neither enjoy nor suffer much, because they live in a gray twilight that knows neither victory nor defeat. (Theodore Roosevelt)

Resumo

A escolha de um processo de desenvolvimento de *software* não é um passo trivial no início de um projeto. Os cenários reais apresentam desafios que nem sempre são abordados na academia. Quando o são, eles se mostram como instâncias particulares de problemas estudados em termos gerais. A aplicação da teoria sem customizações se mostra uma forma ineficaz de executar projetos de *software*. Por isso, este trabalho focará em desenvolver e acompanhar um processo de desenvolvimento que sirva tanto à indústria como a academia. São analisadas algumas teorias psicológicas da formação de equipe e metodologias de desenvolvimento ageis. O objetivo é comparar os resultados da aplicação de um mesmo processo entre uma empresa real e um projeto desenvolvido no âmbito acadêmico. Para tanto é definido um modelo de desenvolvimento customizado capaz de integrar teorias acadêmicas para a solução de um problema real.

Palavras-chaves: Desenvolvimento de *Software*, formação de equipes, gerenciamento de projetos, gerenciamento de projetos ágeis .

Abstract

Choosing a software development process is not a trivial step at the beginning of a project. Real world scenarios present challenges that are not always addressed in academia and when they are, it is as particular instances of problems studied in general terms. The application of sole theory without customizations turns out to be an ineffective way to deal with software development projects. This work will focus on developing and following a such a process. Psychological theories of team formation and agile development methodologies will be analyzed. The objective is to compare the results of a given process between a real company and a project developed in the academic scope. For such it is also defined a customized development model capable of integrating academic theories to solve a real world problem.

Keywords: Software development,team building, project management,agile project management.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BICEN	Biblioteca Central-UFS
CRUD	*CREATE, READ, UPDATE, DELETE*
DCOMP	Departamento de Computação-UFS
IBM	*International Business Machines Corporation*
PETIC	Planejamento Estratégico de Tecnologica da Informação e Comunição
PO	*Product Owner*
RH	*Recursos Humanos*
SaaS	*Software as a Service*
SM	*Scrum Master*
TM	*Team Menber*
UFS	Universidade Federal de Sergipe
XP	*Extreme Programing*

Sumário

1

Introdução

Aplicar o conhecimento acadêmico no mundo real se mostra um problema à parte, afinal "a indústria precisa de engenheiros de software, mas as universidades continuam formando cientista da computação (BECKMAN et al., 1997)". Essa lacuna entre academia e indústria é discutida tanto pelo lado academico (YEN et al., 2003) como pela indústria (ECONOMIST, 2016), (MONSALVE, 2016).

O início deste trabalho deu-se em uma empresa real, sem pretensões acadêmicas. A empresa ACONE[1] procurou em 2014 melhorar seus processos de desenvolvimento, o de Gerente de Projetos, o qual possuia conhecimento gerencial apenas teórico, aplicou diretamente os conhecimentos adquiridos no âmbito acadêmico. Após alguns meses apresentando dificuldades como prazos perdidos, alto índice de retabralho, poucas entregas e alto *turnover* [2], notou-se a necessidade de uma consultoria externa com maior experiência prática. Assim iniciou-se em junho de 2015 uma consultoria com a presença de outro gestor mais experiente em formação de equipes e implantação de metodologias ágeis.

O dia a dia da profissão apresenta particularidades que exigiram uma customização da teoria aprendida em sala de aula. Durante a implantação compromissos foram feitos, erros cometidos, mas ao fim foi possível agregar as teorias acadêmicas em um processo que pudesse fazer o projeto começar a dar certo.

---

[1] Empresa Sergipana que desenvolve sistemas de regulação de saúde pública e caso de uso deste trabalho. https://www.acone.com.br. Mais detalhes serão vistos no capítulo 3.

[2] Rotatividade de pessoas em uma organização.

1.1 Problemas

No âmbito acadêmico encontrou-se o projeto WebPetic[3]. Este demonstrou uma série de problemas em comum com os projetos na ACONE além de um ambiente similar. A academia apresentou um candidato para implantação do mesmo processo estabelecido na empresa ACONE e a comparação dos resultados e melhorias obtidos.

O desenvolvimento de projetos de *software* no âmbito acadêmico apresenta uma série interessante de peculiaridades. Por exemplo a inexistência de vínculo trabalhista, a rotatividade da equipe de pesquisa, a escolha de ferramentas limitada à *softwares* livres ou disponibilizadas através de convênios gratuitos para uso educacional, horários de trabalho flexível e sinergia com o curriculo acadêmico para citar alguns.

1.2 Objetivos e Soluções

Este trabalho de conclusão de curso tem como objetivo geral comparar os resultados da implantação de um mesmo processo de gerenciamento de projetos entre uma empresa e um projeto acadêmico.

Para foram necessarios atingir os seguintes objetivos específicos:(i) Definir uma metodologia de processo de desenvolvimento de *software* que englobasse desde a formação da equipe até gestão diária do trabalho.(ii) Aplicar e documentar os resultados em ambos os ambientes.(iii) Identificar problemas no modelo de desenvolvimento acadêmico e (iv) fazer sugestões de como melhorar o processos.

1.3 Organização do Trabalho

O presente projeto se apresenta dividido em três capítulos além desta introdução. O próximo capítulo versa sobre acerca das possíveis soluções e sua fundamentação, demonstrando quais os métodos e tecnologias capazes de melhorar o processo. No capítulo terceiro serão apresentadas as soluções eleitas e as justificativas. O capítulo final traz à luz o resultado das implantações mencionadas e linhas anteriores e quais as dificuldades encontradas

[3] Projeto do Departamento de Computação da UFS para desenvolver uma ferramenta web para criação de artefatos PETIC. Mais detalhes serão vistos no capítulo 3.

2

Fundamentação Teórica

Este capítulo apresenta conceitos essenciais que serviram de alicerce teórico para o desenvolvimento deste projeto de pesquisa. Aqui são apresentados o Framework Petic, metodologias de projeto e tecnologias possíveis para a implementaçao do sistema. Ao final, descreve-se o ambiente dos estudos de caso.

A revisão bibliográfica foi feita através da busca de artigos com as palavras chave de *"team building", "team training", "team forming" ,"agile coaching", "agile project", "agile project management", "scrum virtual teams"*, Esses termos foram buscados nos repositórios acadêmicos digitais fornecidos pela UFS e Google *Scholar*. Alguns artigos encontrados através do Google *Scholar*, indisponíveis nas bases da UFS, foram adquiridos através de repositórios de acesso pago à conteúdo acadêmico. Também foram consultados livros próprios e disponíveis na Biblioteca Central da UFS, BICEN, sobre os tópicos de Formação de equipes, *Scrum* e Gerência de projetos.

Faz-se relevante lembrar que durante a implantação na empresa ACONE existiu acesso à consultoria especializada para gerenciamento de equipes e implantação de metodologias ágeis, principalmente *scrum*, na empresa. Sendo assim, o trabalho foi inicialmente direcionado à uma bibliografia voltada para a parte prática de criação de equipes ágeis.

2.1 Formação de equipe

A formação de equipes é um tópico inicialmente estudado na área da psicologia. Os trabalhos escritos sobre formação de equipe não se limitam apenas a uma área de atuação, o resultado do estudo de suas técnicas podem ser encontrados em áreas diversas e demonstrando "exuberantes"resultados , Construção civil (OWEN et al., 2006), Vendas (Vick,2008), Aviação civil (NTSB,2010) e claro desenvolvimento de *software* (SCHARFF; VERMA, 2010),(HIGHSMITH;

Tabela 1 – Os quatro componentes de intervenções na formação de grupos de Beer & Buller

Intervenção	Resultado esperado por Beer & Buller
Atribuição de metas	Enfase em definição dos objetivos e desenvolvimento de metas individuais e de time. Membros da equipe devem ser envolver no planejamento das ações para identificar maneiras de para atingir as metas.
Relações interpessoais	Membros do time devem desenvolver lealdade entre os menbros e confiança no time.
Resolução de problemas	Membros do time devem se envolver no planejamento da solução de problemas e na avaliação das soluções propostas
Definição de atribuições	Enfase na clara comunicação aos membros do time sobre suas responsabilidades e de seus pares dentro da equipe

Fonte: Beer,1976 & Buller,1086

COCKBURN, 2001)

O método científico, no entanto, nos obriga a duvidar, questionar e analisar os resultados obtidos de maneira imparcial. Esses questionamentos são abordados por Salas et al. (1999) em seu meta-estudo sobre os efeitos de intervenções de formação de grupos nas medidas de performance. O estudo examina 35 artigos acadêmicos sobre os resultados obtidos em relação as intervenções de formação de equipe e como esses componentes se relacionam ao tamanho das equipes, duração das intervenções e tipo de medidas utilizadas (subjetivas x objetivas).A intervenções analisadas são agrupadas em quatro componentes de intervenções na formação de grupos segundo Beer (1976), Buller (1986) :(i) Atribuição de metas,(ii) relações interpessoais,(iii) resolução de problemas e (iv) definição de atribuições ,ver Tabela 1.

Os resultados do meta-estudo sobre o material publicado à época atestou que o efeito geral das intervenções é insignificante (SALAS et al., 1999). Quando analisado individualmente o componente definição de atribuições, este foi o único que apresentou uma diferença na performance subjetiva do projeto. A efetividade das intervenções diminui conforme o número de membros na equipe aumenta. Notou-se também uma tendência a diminuição da efetividade conforme a duração das intervenções aumenta. Por fim, Salas et al. (1999) não encontra nenhuma alteração nas medidas objetivas de performance.

Ainda buscando entender à disparidade entre os resultados sobre a efetividade das intervenções de formação de grupos os estudos na area continuaram. Em 2011 Shuffler diz:“ Todos os quatro componentes tiveram efeitos moderados na performance, sendo Atribuição de metas e Definição de atribuições os mais relevantes” (SHUFFLER; DIAZGRANADOS; SALAS, 2011)

Nessa revisão sobre os resultados da eficiêcia de intervenções para o desenvolvimento de equipes é notado que “...Existe uma ciência por trás da formação de equipes eficientes, no entanto a ciência deve ser aplicada- e aplicada corretamente- para o sucesso do desenvolvimento das equipes.”. A utilização de técnicas desenvolvidas fora do método cientifico pode levar desde

Tabela 2 – Os seis elementos chave da formação de equipe

Intervenção	Definição da intervenção por Haggen
i	Em todas as ações demonstrar respeito e consideração pelos funcionários como valiosos membros da equipe.
ii	Identificar responsabilidades individuais e níveis de performance esperados. Garantir que todos os membros da equipe estão cientes da importância individual de seus trabalhos.
iii	Garantir uma comunicação clara entre funcionários como indivíduos e como time. Valorizar opiniões e feedback dos e para os funcionários.
iv	Estabelecer metas individuais e de time, de preferencia em coordenação com os envolvidos. Considerar igualmente a possibilidade de crescimento profissional.
v	Recompensar trabalho em equipe e fortalecimento do time. Deixar claro para equipe quem é o responsável por distribuir as recompensas e quais tipos de recompensa são oferecidos.
vi	Praticar a lealdade entre membros da equipe. Fomentar um ambiente de confiança entre membros e garantir que os supervisores diretos irão defender a equipe contra injustiças.

Fonte: Hagen (1985)

simples resultados negativos na produtividade até a processos judiciais por conta de técnicas inapropriadas (VICK, 2008). ou perca de vidas quando uma equipe falha em um momento crucial como a queda de um avião (NTSB, 1979).

Já na abordagem de Hagen (1985) existem seis elementos chave na formação de times.

> (i) respeito e consideração. (ii) Identificar expectativas e responsabilidades. (iii) comunicação entre menbros,(iv) metas individuais e de grupo. (v) recompensa por trabalho de equipe e fortalecimento do time. (vi) encorajar lealdade e confiança no time.

Cada um desses topicos é, por si, explorado em outros artigos(TIPPETT; PETERS, 1995),(TERRES; SANTOS, 2010). É possivel detectar uma sobreposição de escopo na definição dos componentes entre os principais autores. Assim, sem a intenção de criar ainda outra classificação com diferença meramente semântica consideraremos seus princípios na formulação do processo a ser adotado.

2.1.1 Confiança em times virtuais

"Confiança entre os membros de uma equipe é imperativo para a sucesso de um projeto ágil" (DORAIRAJ; NOBLE, 2013)

Dentro dos fatores desejáveis a um time desenvolvimento da confiança se destacou como um assunto recorrente na bibliografia, especialmente para times virtuais e ágeis. Novamente existe uma sobreposição de conceitos em relação aos já mencionados. A questão dos artigos

Tabela 3 – Fases de formação de equipe por Tuckman

Fase	Definição
Formação	Produtividade é baixa por ser uma fase de exploração da nova situação. O foco é montar a equipe , estabelecer objetivos e responsabilidades,interação entre os membros e comprometimento.
Conflito	Os problemas interpessoais começam a aparecer e se refletir nos resultados. O grupo começa procurar "culpados" a cultura do grupo também começa a ser definida e os papeis devem ser reforçados
Normatização	A resistência a formação do grupo abaixa e novos padrões de comportamento são definidos pelo próprio grupo. As opiniões pessoais sobre as atividades são expressadas.
Performance	O grupo começa a aceitar as tarefas como impessoais. Os papeis são flexíveis e funcionais, a estrutura do grupo tem como objetivo entregar as tarefas e entender a relação entre clientes e gerentes como parceiros de um objetivo em comum.
Desintegração	Para equipes temporárias é o momento de transição para a próxima equipe e tranquilizar os membros sobre o processo futuro

Fonte: Tuckman (1965)

se apresenta acerca de como criar esse ambiente de confiança entre programadores, analistas, representantes do cliente e *scrum masters*[1] de maneira a permitir que a comunicação seja fluente.

O primeiro passo para desenvolvimento de confiança é estabelecer canais de comunicação claros que possam iniciar uma conversação, a qual irá gradualmente melhorar a capacidade da equipe de debater assuntos em profundidade de maneira aberta (HOLTON, 2001).

Existiram dúvidas sobre a capacidade real de um time virtual ser capaz de atingir níveis de produtividade semelhantes aos que interagem diariamente de maneira presencial. No entanto, pesquisas demonstram que:

> ... times virtuais conseguem chegar a consensos mais rápido, a conversa não é dominada por nenhum membro e em geral exibem mais comportamento verbal (Kling, 1995).

Confiança, além dos fatores visto previamente, respeito, atribuição de metas ao time, incentivar a colaboração em atividades, evitar aparecimento de "donos de produto" [2] e deixar que o time faça escolhas sobre seu próprio gerenciamento, criam o espírito de cumplicidade necessario à formação de um time.

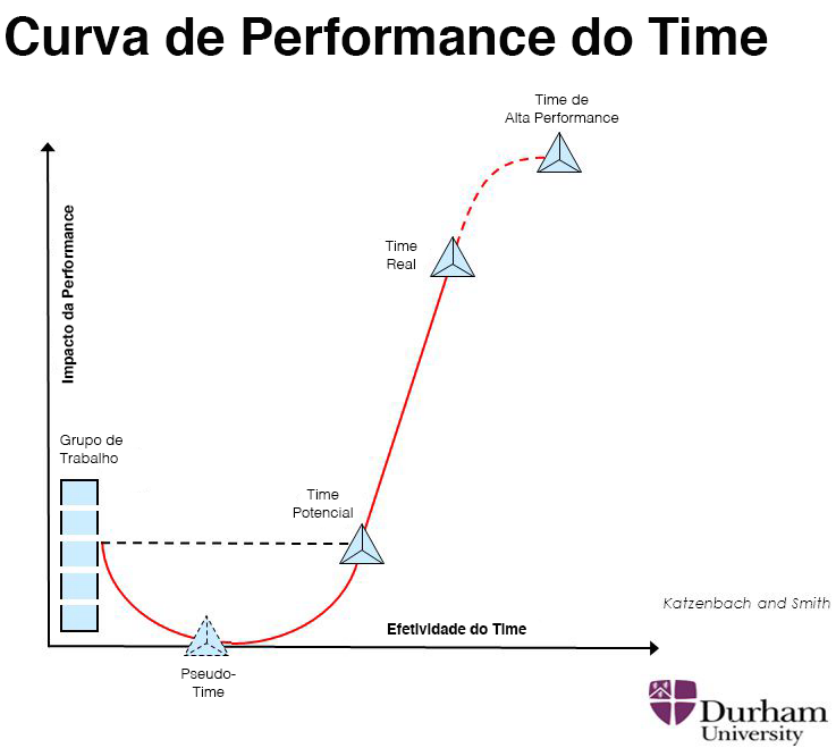
2.1.2 Fases da Formação de equipe

De acordo com (TUCKMAN, 1965), existem 5 fases predominantes na formação de equipes: (i) Formação, (ii) Conflito, (iii) Normatização, (iv) desempenho e (v) desintegração.

[1] Título dado à profissionais responsáveis por garantir a devida execução do processo Scrum.

[2] O termo donos de produto se refere à pratica de programadores não compartilharem a expertise relacionada ao código de um *software* dificultando a manutenção por terceiros. Não confundir com PO da metodologia *scrum*

Figura 1 – Performance X Maturidade.



Fonte: Fonte: Hannaway (2016).

Que podem ser, resumidamente, definidas como visto na Tabela 3

Essa fases de formação de equipe são diretamente ligadas à produtividade (KATZENBACH; SMITH, 1993), enquanto em um momento inicial pode existir uma queda na performance conforme visto na Figura 1, esta é seguida de um aumento substancial. É importante deixar os stakeholders cientes desse comportamento, para que a implantação de intervenções de formação de equipes não sejam abortadas prematuramente.

2.2 Processo de desenvolvimento de *software*

De acordo com Sommerville (2010) podemos apontar 4 processos genéricos que são a base para a construção de *software* : (i) especificação de *software*, (ii) design e implementação de *software*, (iii) validação de *software* e (iv) evolução do *software*.

- Especificação de *software*: As funcionalidades do *software* e limitações de sua operação devem ser definidos.
- Design e implementação de *software*: O *software* deve ser desenhado e produzido.

- Validação de *software*: O *software* deve ser validado para garantir que atende as necessidades do cliente.
- Evolução do *software*: O *software* deve evoluir para atender as mudanças das necessidades do cliente.

Esses processos são, por si só, um conjunto complexo de atividades. Porém todos os processos atuais, (watterfall, ágil/iterativo, RUP, espiral), de alguma maneira implementam esses passos. Em seguida fornecemos uma breve revisão dos processos conforme definido por Sommerville (2010)

2.2.1 Especificação de *software*

Este processo contempla a definição de requisitos do *software*, bem como a identificação do ambiente em que o mesmo será utilizado e quais as limitações de sua utilização. O objetivo dessa fase é entender qual problema o cliente precisa ter resolvido, quais os requisitos à serem atendidos, quais as limitações de escopo e do ambiente e analisando, ao final, se da fato o *software* deve ou não ser produzido.

Tal processo é conhecidamente crítico no desenvolvimento de *software*, já em 1986 , Boehm (1988) apontava que

> " uma grande parte do esforço em projetos de *software* são dedicados a retrabalho (...) necessário para compensar por requerimentos mal definidos ou erros na especificação (...) seu custo chega normalmente a 50% do orçamento de grandes projetos".

Além do custo financeiro existe o desgaste das relações cliente-fornecedor e o eventual atraso ou cancelamento do projeto.

2.2.2 Arquitetura e implementação de *software*

Aparentemente são dois processos separados. Porém será visto na seção 2.3 como arquitetura e implementação são considerações de uma mesma fase de projeto (Execução). Tendo como entrada nesse processo os requisitos levantados no processo anterior, seu objetivo é produzir um *software* que os atendam. Para isso, temos a produção de uma arquitetura do sistema, desenho de interfaces, modelos de banco de dados e arquitetura de componentes. Então, a partir dessas sáidas intermediárias é desenvolvido o sistema.

2.2.3 Validação de *software*

Apesar de ser um processo normalmente mais utilizado na fase final, quando da aceitação do produto, existe também validação de *software* durante todo o processo de desenvolvimento. A

Figura 2 – Custo de correção de defeitos.

Fonte: Briski (2008)

validação dos requerimentos contribui para mitigar os problemas citados na fase de especificação, enquanto testes de desenvolvimento e integração apresentam *feedback* para a fase de arquitetura.

Nos testes de aceitações temos a finalização do processo de desenvolvimento e o *software* é entregue para o uso do cliente. O sistema é testado com dados reais do cliente. Caso os processos de especificação e implementação tiverem produzidos defeitos, esse é o momento que a maioria deles será descoberto.

É importante notar que conforme o desenvolvimento avança, os testes ficam mais longos, complexos e a resolução de erros mais custosa, conforme podemos observar na Figura 2.

2.2.4 Evolução de *software*

Processo comumente traduzido para o português como "Manutenção de *software*", este consiste em processos que visam garantir que o *software* entregue irá receber correções e implementar novas demandas de acordo com a necessidade do cliente. É valido ressaltar que, evolução e suporte a operações são processos diferentes. O último se refere à suporte da instalação e utilização do *software* pelos clientes, sem geração de demanda para o desenvolvimento. Evolução é adição de novas funcionalidades afim de atingir requisitos além da concepção inicial.

Todavia segundo Sommerville (2010) , " ...a distinção entre implementação e evolução se torna cada vez mais irrelevante ..." O que é premissa cada vez mais comum em processos incrementais de *software* e *software* como serviço Slate (2009).

2.3 Modelos de Processo

2.3.1 Cascata ou Watterfall

O modelo em cascata ou linear tem esse nome por que as saídas de cada etapa do processo são despejadas para as seguintes, como uma cascata. É um exemplo de processo voltado para o planejamento, todas as atividades precisam ser definidas antes que a execução seja iniciada. Em Termos genericos suas principais fases são : (i) Analise e definição dos requerimentos, (ii) Arquitetura do *software*, (iii) Implementação e teste unitarios, (iv) integração e validação e após sua entrega (v) suporte e operação.Sommerville (2010).

A definição dessas fases seguem o conceito visto anteriormente. O que diferencia o processo em cascata dos demais é a maneira como essas fases interagem entre si e com o cliente. A principio, cada fase é detalhadamente documentada, suas saídas são aprovadas e assim iniciada a próxima fase. Caso seja detectado um erro na fase anterior, todo o projeto retorna à mesma. Isso também é verdade caso exista alguma requisição de mudança, pois todo o projeto deve ser revisto para encaixar os novos requerimentos. Além disso, a natureza linear atrapalha o paralelismo do trabalho, regularmente levando o projeto a bloqueios (BRA 1994). As principais críticas desse modelo são:

> (i) Na prática poucos projetos reais seguem de fato um sequencia linear tão bem estabelecida.(ii) O processo não é aberto a alterações ou feedback do cliente.(iii)É necessário que o cliente possa definir claramente todos os seus requisitos inicialmente. (iv)Principalmente o produto é entregue de uma única vez, caso um erro de requerimento seja detectado nessa fase, a necessidade do cliente mude ou o ambiente se altere o produto pode se tornar obsoleto ou no melhor caso, passível de um longo e custoso retrabalho Pressman (2005).

No entanto, esse modelo ainda é indicado quando do desenvolvimento de aplicação críticas. O escopo é completamente conhecido desde o início do projeto e de fato não é possível entregar um produto com funcionalidade parcial. Um exemplo de sucesso desse modelo são projetos espaciais. A sonda de marte *Curiosity* é um projeto de *hardware* e *software* que possuía um escopo muito bem definido e no qual uma falha em produção seria irreparável. A sonda foi lançada em 2011 com sensores de câmera escolhido em 2004, na época de lançamento, muito inferiores a qualidade de câmera de celular. Quando questionada por que o projeto não foi alterado para incluir sensores mais modernos a equipe da NASA respondeu que "... nós já tinhamos dados de teste de radiação suficiente [para o sensor]...". Logo alterar o sensor escolhido em 2011 teria levado a todo um retrabalho e reteste do projeto. Esse tipo de comportamento do processo, em não alterar os requerimentos, pode ser apontada como responsável pela sonda continuar funcionando por quase o dobro do tempo inicialmente esperado de dois anos (DPREVIEW, 2012),(SPACE 2012).

2.3.2 Incremental ou Evolucionario

O processo Incremental segue os mesmos passos de processos já vistos anteriormente. Seu diferencial é adotar vários ciclos desse processo cada um com seu escopo reduzido. A cada ciclo uma parte do produto é desenvolvida e entregue. O cliente, além de receber valor imediato, pode fornecer feedback ao processo e apontar quais os requerimentos mais importantes, para o próximo ciclo. Mudanças são acomodadas sem grande necessidade de retrabalho e ,caso os requerimentos apresentem algum defeito, apenas o ciclo atual terá sido comprometido. A percepção de progresso pelo cliente é mais clara, uma vez que, pode verificar o funcionamento do produto.

Quando o escopo não está totalmente definido, ou quando o ambiente em que o produto será inserido muda com frequência, o modelo incrementral é indicado.

Apesar das vantagens do processo incremental, existem limitações para a sua implementação. Como visto, esse processo não é indicado para o desenvolvimento de *software* crítico. Sistemas grandes e complexos exigem uma arquitetura estável. Conforme o sistema evolui existe a possibilidade de que sua arquitetura comece a receber incrementos que não seguem padrões estabelecidos de *software* para alocar novos requerimentos.

Outro problema é sobre a dificuldade de precisar o progresso do projeto. O gerenciamento de projetos tradicional trabalha continuamente com a alocação de recursos. Se um time está alocado a um projeto é necessário saber por quanto tempo aquele recurso será consumido no projeto(SOMMERVILLE, 2010). Como os requerimentos são analisados e definidos a cada ciclo definir o prazo final para a entrega do projeto se torna um problema a parte.

Um caso de sucesso que conseguiu lidar com as dificuldades do processo incremental para um grande sistema é visto no programa SENTINEL[3]. Depois de 10 anos tentando substituir o seu sistema de informação através de um processo em cascata, que entregou produtos atrasados, caros e que não entregavam valor (MARCHEWKA, 2010), o FBI decidiu utilizar uma abordagem incremental. Os requerimentos monolíticos foram quebrados em 670 histórias de usuário que foram priorizadas e entregues em ciclos de duas semanas (WERNHAM, 2012).Nessa nova abordagem mais de 80% dos requerimentos foram entregues no primeiro ano e apenas 50% do orçamento total tinha sido utilizado.

2.3.3 Metodologia Ágil

> Estamos descobrindo maneiras melhores de desenvolver *software*, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho,

[3] Software de gerenciamento de casos desenvolvido pelo US FBI com o objetivo de substituir os processos de papel por um fluxo completamente digital

> passamos a valorizar:
> Indivíduos e interação entre eles mais que processos e ferramentas;
> *software* em funcionamento mais que documentação abrangente;
> Colaboração com o cliente mais que negociação de contratos;
> Responder a mudanças mais que seguir um plano;
> Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda Beck et al. (2001).

Métodos ágeis são "...processos incrementais , aonde adições pequenas adições são feitas a cada iteração e disponibilizadas para o cliente..." (SOMMERVILLE, 2010). Assim, herdando e expandindo características do modelo incremental temos : (i) Especificação, Arquitetura e Produção intervalados, Documentação é reduzida ao mínimo necessário; (ii) Sistema desenvolvido em versões entregues regularmente, com forte envolvimento do cliente; (iii) Intefaces produzidas interativamente como protótipos de tela.

A metodologia ágil potencializa os prós do processo Incremental e adiciona seus próprios contras ao modelo. A documentação reduzida pode criar um futuro problema de manutenção do sistema, ou problemas quando novos membros são incluídos na equipe. A necessidade de um representante do cliente com alta disponibilidade e a dificuldade de formatar contratos de aquisição com escopos não completamente definidos. Detalhes do modelo ágil serão abordados na discussão de *scrum*.

2.3.3.1 Cascata, Incremental e Ágil

Além dos prós e contras levantados sobre os modelos é importante comparar os resultados das suas aplicações. O CHAOS report (2015) nos traz alguma informações consolidadas sobre a performance relativa de cada modelo na Figura 3. Essa abordagem inclui incremental agregado como um subgrupo de Ágil. O relatório é fruto do levantamento de dados em 50 mil projetos de *software*. Pode se constatar indices de sucesso superiores ao modelo cascata para todos os tamanhos de projeto, inclusive em projetos de grande porte, indo de encontro às sugestões acadêmicas.

2.3.4 Métricas

Como observado no artigo de Salas et al. (1999) é preciso acompanhar os indicadores para poder apontar se as técnicas utilizadas estão de fato trazendo melhorias ao processo. No entanto, a pergunta feita para o nosso ambiente é o que medir e como medir. (HARTMANN, 2006) apresenta em seu artigo considerações sobre como escolher métricas e diagnósticos voltado para entrega de valor ao usuário. Contabilizar o projeto através do trabalho realizado (Pontos de função, Linhas de Código) não garante a entrega de um produto com qualidade nem de um produto que satisfaça o cliente. Em contraponto muitas métricas comuns aos projetos ágeis (Velocity, Story points, Bussines value) são voltadas para o valor entregue. Estas, por si só, não

Figura 3 – Status dos projetos FY2011-2105, um comparativo entre Agil(incremental) e cascata(waterfall).

Resolução do Chaos por Ágil vs Cascata

Tamanho	Metodologia	Bem Sucedido	Desafiado	Falho
Todos os tamanhos	Ágil	39%	52%	9%
	Cascata	11%	60%	29%
Grandes Projetos	Ágil	18%	59%	23%
	Cascata	3%	55%	42%
Projetos Médios	Ágil	27%	62%	11%
	Cascata	7%	69%	25%
Pequenos Projetos	Ágil	56%	38%	4%
	Cascata	44%	45%	11%

A resolução de todos os projetos de softwares da FY2011-2015 contidas no banco de dados CHAOS. Segmentada por processo ágil e método cascata. O total de de projetos supera 10.000.

Fonte: Hastie (2015)

garantem a “qualidade” do produto, mas a sua adequabilidade para atingir o objetivo desejado. O foco em valor é uma premissa comum em projetos ágeis.

2.4 *scrum*

Scrum é um framework ágil de gerenciamento e controle de projetos. Seu foco é direcionado ao gerenciamento das atividade e não na execução das tarefas em si. Por ser um framework ,apesar de comumente utilizado na área de desenvolvimento de *software* é utilizado com sucesso em projetos de engenharia civil.

O processo de adotar *scrum* segundo Somerville (2010) consistem em 3 fases:"(i) Definição dos objetivos gerais do projeto e definição da arquitetura do *software*, (ii) Series de ciclos (chamados de *sprints* na metodologia) e (iii) encerramento do projeto."

Em comparação com os processos vistos anteriormente, não existe um passo dedicado exclusivamente para testes e validações, estes ocorrem durante as *sprints*. Não existe também um planejamento inicial de todo o projeto, mas para evitar os problemas de qualidade e cobertura que costumar afetar os projetos incrementais, a arquitetura do *software* é definida inicialmente para dar suporte aos requerimentos.

2.4.1 Papéis

Existem 3 papeis no *scrum*: o Dono do Produto (PO), *scrum* Master (SM) e o membro da equipe (TM). O dono do produto tem como responsabilidade: representar os interesses do cliente, organizar as prioridades do backlog do produto, quais itens devem ser retirados e inseridos no backlog do produto, garantir que os requisitos do cliente estejam claros e aceitar ou rejeitar as entregas de cada sprint.

O *scrum* Master tem como responsabilidade: auxiliar o devido entendimento e execução das regras, processos e teoria do processo estabelecido. Bem como auxiliar o PO e TMs nas suas responsabilidades o *scrum*. Também é de responsabilidade dele remover impedimentos ao andamento das atividades, facilitar as cerimonias do *scrum* e proteger o backlog da *sprint* contra alterações desnecessárias. Membros da equipe, normalmente alocados em times que variam entre 3 e 9 pessoas, são os responsáveis por produzir o *software* de maneira atender o backlog da *sprint*. O time é completamente auto gerenciável, e não cabe a ninguém interferir na organização ou execução do trabalho.

2.4.2 Cerimônias e Artefatos

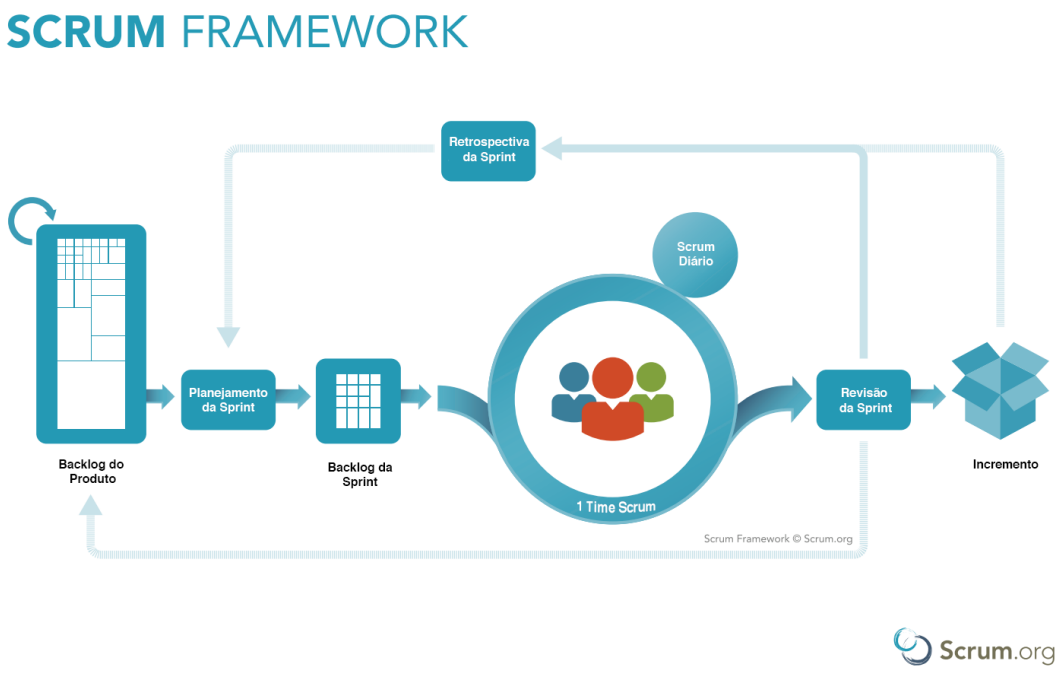
Existem principalmente 6 artefatos no *scrum*: (i)backlog do produto, é uma lista de requisitos que devem ser atendidos pelo sistema, (ii)*backlog* da *sprint*,requisitos selecionado do *backlog* do produto que serão desenvolvidos durante a *sprint*, também encarada como meta da *sprint* e (iii)incrementos, os componentes de *software* que atendem requisitos e foram produzidos e aceitos durante e *sprint*, (iv)definição de pronto, entendimento comum do que é necessario para uma tarefa ser considerada completa, (v)monitaramento do progresso da *sprint*, acompanhamento de performance e previsão de desempenho de uma *sprint*, (vi)monitoramenteo do progresso de objetivo, acompanhamento de performance e previsão de desempenho de todo o projeto.

A metodologia define que as cerimônias e as fases do processo devem ser limitadas pelo tempo (*Time-boxed*), ao invés de um escopo definido as cerimonias tem um tempo definido e é preciso atingir um resultado dentro desse tempo. O modelos sugerido pelo *scrum* pode ser visto na figura 4.

Sprints são onde o desenvolvimento do produto acontece de fato, seu objetivo é entregar um pedaço de incremento potencialmente utilizáveis para o cliente. Elas possuem um tempo pré-determinado que engloba também as cerimônias de Planejamento, *scrum* diário, Revisão e Retrospectiva. Durante a *sprint*, não devem ocorrer mudanças que possam por em risco a meta ou alterar a qualidade do produto definido. Impedimentos devem ser removidos e o PO deve estar disponível para esclarecer dúvidas sobre o escopo.

O Planejamento da *Sprint* é a reunião que decide quais tarefas do backlog do produto serão produzidas. Essa escolha é negociada com todos os membros envolvidos no processo (PO, SM e TM). O objetivo popular o backlog da *Sprint* e como as tarefas serão organizadas para

Figura 4 – Cerimonias e Artefatos do *scrum*



Fonte: (SCRUM.ORG, 2016)

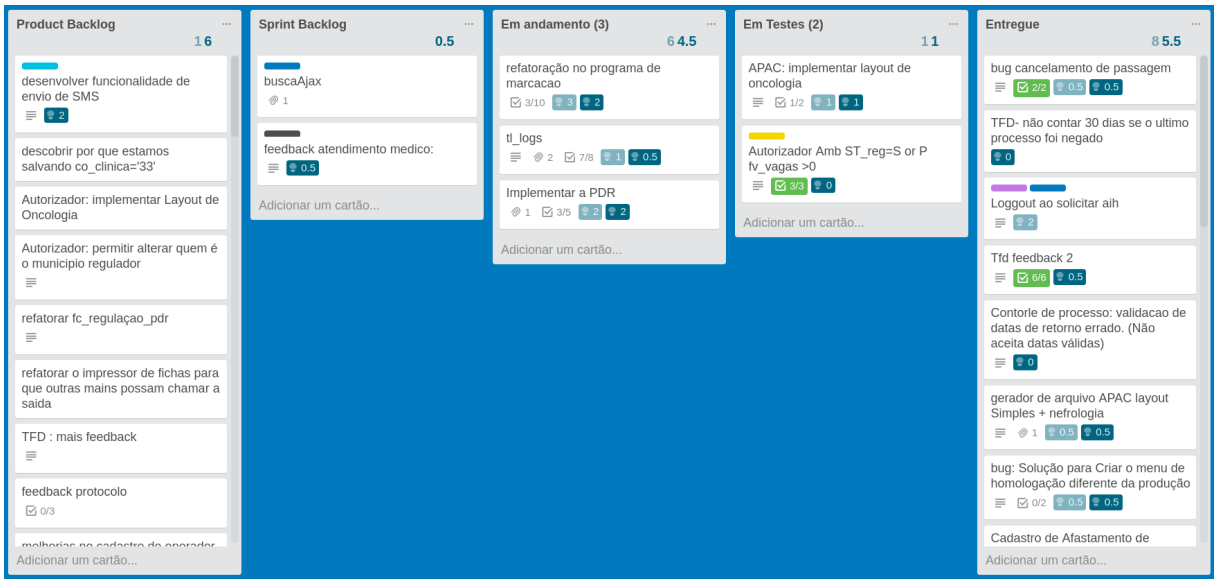
atender a meta estabelecida. O tempo varia de 2h a 8h para essa cerimonia

A *Scrum* diária consiste em uma reunião entre os membros da equipe (SM e TM) no qual 3 questões são respondidas por cada TM: 1- O que eu fiz desde a última *scrum* diária para atingir a meta da *sprint*, 2- O que farei até a próxima *scrum* diária para atingir a meta da *sprint* e 3- Existe algum impedimento às atividades que possa impedir o time de atingir a meta da *sprint*? Essa cerimônia deve durar no máximo 15 minutos. Logo após o SM deve proceder o removimento dos impedimentos.

Na Revisão da *sprint*, todos os membros envolvidos no processo se reúnem novamente e os TMs demonstram o produto para o PO. São debatidos quais problemas ocorreram e quais funcionalidades foram entregues e como a entrega desses incrementos afeta o desenvolvimento do produto como um todo. Esse processo deve demorar de 2h a 4 h.

Retrospectiva é o momento que todos os membros da equipe discutem o andamento da *sprint* que passou. O objetivo é tentar melhorar continuamente o processo. São analisados processos de desenvolvimento, relação entre os membros, ferramentas utilizadas, itens que podem ser melhorados e o que deixar para trás. Por fim, um acordo de como implementar melhorias ao processo é definido. Essa cerimonia leva de 1h a 3 h.

Figura 5 – Quadro Kanban da empresa Acone



Fonte: Produzido pelo Autor

2.5 Técnicas relacionadas

Kanban, XP e *Planning Poker* são técnicas que podem interagir de maneira sinérgica com *scrum* definindo como implementar atividades de baixo nível. Sua utilização é tão comumente vinculada à metodologia *scrum* que podem ser erroneamente interpretadas como parte da mesma.

2.5.1 *Kanban*

Kanban é uma técnica visual de gestão de fluxo, do japonês "sinal visual", tem como origem a produção *just in time*[4] do Toyotismo (JMA, 1986). Como visto em (RAD, 2014) Seu objetivo é deixar claro para os menbros da equipe quais tarefas existem e qual o estado de cada uma delas.

Tarefas estão sempre em um determinado estado. Cada estado possui um limite de tarefas associadas. Na Figura 5 podemos ver um exemplo do quadro *Kanban* da ACONE. As raias representam o estado das atividades no fluxo de produção, o valor entre parênteses indica a quantidade máxima de tarefas associadas aquele estado. No exemplo vemos a implementação do backlog do produto, o backlog da *sprint* e o fluxo dos cartões de tarefa durante a *sprint*.

A decisão de limitar as atividade em cada estado tem como objetivo implementar um sistema em que as atividades são "puxadas". Por exemplo, ao atingir um limite de 2 tarefas em testes, novas tarefas não podem ser empurradas para a raia de testes até que um espaço seja liberado, forçando a equipe a trabalhar para resolver o gargalo.

[4] JIT um sistema de administração da produção que determina que tudo deve ser produzido, transportado ou comprado na hora exata. Pode ser aplicado em qualquer organização, para reduzir estoques e os custos decorrentes.

Considerando a limitação de trabalho em cada estado a empresa precisa gerenciar a mão de obra relacionada a cada estado. Em uma equipe tradicional de desenvolvimento de *software* isso significaria adequar os números de analistas, programadores (*back end & front end*) e testadores afim de estabelecer um fluxo sem gargalos ou ociosidade.

2.5.2 XP

XP é uma metodologia de desenvolvimento ágil que contempla tanto o processo de gerenciamento, como as práticas a serem utilizadas. Como foi escolhido *scrum* para framework de gerenciamento do projeto, a parte a ser utilizada do XP são as práticas. Não é sugerido pela literatura que todas as sejam executadas, mas que cada organização escolha as praticas que lhe convém. Para a este trabalho foi decidido utilizar essas práticas no dia-à-dia da *sprint* :

Planejamento incremental: XP define a utilização de cartões de historias de usuário para definir os requisitos desejados. Esse cartões são criados durante a *sprint* pelo PO e colocados no Kanban, que serve como itens do backlog *scrum*.

Entregas Incrementais: XP sugere que sempre que uma funcionalidade é desenvolvida deve ser integrada ao produto final. Foi decidido seguir o ciclo do *scrum* para enviar aos clientes finais, porem todas as funcionalidades são integradas em um ambiente de ambiente de homologação.

Arquitetura simples: apenas o desing requerido para atender o requisito atual é definido, nenhuma previsão para funcionalidades futuras é incluido.

Código Coletivo : Apesar do XP sugerir programação em pares o ambiente virtual dos times se prova um empecilho ao mesmo, assim, essa pratica é substituída em revisão por pares que ocorre durante a fase de testes dos cartões.

Ritmo sustentável: Horas extras não são consideradas como uma maneira eficiente de atingir as metas,pois a longo prazo diminui a produtividade e pode reduzir a qualidade de código.

2.5.3 *Planning Poker*

Poker de Planejamento é uma técnica que consiste em pontuar qual o tamanho do esforço para desenvolver um determinado produto, a partir de uma régua arbitraria. Essa pontuação é dada pelos TM para as tarefas que irão compor o *sprint backlog*. Após esclarecimento de um cartão, os membros do time pontuam, individualmente, quantos eles acreditam que será necessario se dedicar a uma tarefa. As estimativas são reveladas e a atividade é discutida, o processo se repete até que seja encontrado um consenso.

3

Processo de Desenvolvimento Definido

Nessa seção explicitaremos brevemente o ambiente dos estudos de casos selecionados. Descreve-se a situação da empresa ACONE e do projeto WebPetic e como foi realizado o acompanhamento das atividades. Por fim são comparados os ambientes e proposta uma solução.

3.1 Estudos de Caso

3.1.1 ACONE

Fornecedor de sistema de regulação de consultas e procedimentos para saúde pública desde 1999. Trabalha com um modelo de SaaS[1] para disponibilizar o acesso ao SUS. Possui na sua carteira de clientes, todas as regionais de saúde do Estado de Sergipe, Aracaju, Estância, Nossa Senhora do Socorro, Itabaiana, alem da própria secretaria de saúde estadual. O sistema atende a população mais carente desses municipíos, e de municipíos menores atendidos pelas regionais. São processados em média 30 mil exames e consultas a cada dia útil.

Atualmente a estrutura da empresa conta com três programadores , um gerente de projetos e um responsavel por infraestrutura, além do setor administrativo. Em 2014 o processo de desenvolvimento e gerenciamento dos projetos era bastante incipiente. As demandas eram organizadas em uma lista de atividades que a equipe assumia conforme acabavam as demandas em que estavam alocados. Os prazos para cada atividade eram definidos pelo desenvolvedor junto ao gerente de projetos. A equipe distribuída em Sergipe e outros estados, trabalhava de maneira virtual. Reuniões irregulares eram realizadas através de videoconferência e a comunicação no dia-a-dia se dava através de e-mail.

[1] Modelo de distribuição de *software* aonde o fornecedor do *software* se responsabiliza por toda a estrutura necessária à disponibilização do sistema (servidores, conectividade, cuidados com segurança da informação)

Como resultado desse processo a equipe passou por uma série de problemas, prazos perdidos, métricas de produtividade inexistentes, insatisfação da alta gerência com o time e alta rotatividade da equipe entre os principais. Sem a possibilidade de prever a produtividade da equipe, se fez impossível gerenciar um cronograma de projetos, definir quando os programas seriam entregue para o usuario ou mesmo quanto exatamente cobrar por uma funcionalidade.

O ambiente em que a empresa está inserida também apresenta seus próprios desafios. A falta de organização de processos para o atendimento de saúde pública sempre foi um problema para a empresa, sem uma visão organizada de como trabalhar com os clientes que sempre demandaram mudanças. As instabilidades políticas, no período de 2014 à 2016, levaram a uma alta rotativa dos gestores públicos. Cada novo gestor assumindo suas atividades sem uma visão de continuidade do plano iniciado pelo seu antecessor. Projetos foram encerrados e mudança de requerimentos ocorreram com frequência, exigindo do processo uma capacidade de se adaptar à mudanças que este não possuía.

Nesse ambiente, se fez necessário repensar o processo de desenvolvimento da ACONE, considerando as particularidades do projeto, a disponibilidade de treinamento e assessoria técnica disponível.

É relevante comentar sobre os processos que existiam previamente e foram consideradas improdutivos, como referência para que possam aprender a partir de nossa experiência.

Reunião diária assíncrona: Inicialmente devido à dificuldade de encontrar um horário compatível para todos os TMs a empresa utilizava um processo em que cada desenvolvedor enviava ao final do seu "dia de trabalho" um email respondendo as 3 perguntas da reunião diária. Esse processo podia deixar tarefas bloqueadas por até um dia inteiro, não facilitava a evolução das relações interpessoais nem o auto-gerenciamento do time.

Estimativas de tarefas sem lastro: No processo de estimativa prévio, cada programador se reunia com o gerente do projeto e a tarefa era explicada em alto nível. A ausência de documentação de requisitos e prototipos de tela permitia um alto nivel de incerteza. Via de regra existia retrabalho nas tarefas e consequentemente a perda de prazos

Levantamento de requisitos distante do usuario final: Questões politicas interferiram na interação com o maior cliente da empresa, que passou a demandar a seus requisitos através de um gestor. A informação entre o usuario final e equipe de desenvolvimento passava por pelo menos três setores administrativos. Causando assim perca de informação e dificuldade em priorização das tarefas.

3.1.1.1 Metodologia de acompanhamento

As atividades de treinamento da equipe foram realizadas através de conferências *online*, registro quinzenal das métricas de produtividade da *sprints* e acompanhamento das cerimônias de abertura,retrospecitiva e encerramento das *sprints*.

3.1.2 O projeto WebPETIC

A metodologia PETIC, Planejamento Estratégico de Tecnologia de Informação e Comunicação, é um framework para definição do planejamento estratégico de organizações, idealizado e desenvolvido primariamente no DCOMP da Universidade Federal de Sergipe.

O sistema Web-Petic se propõe a ser uma ferramenta que auxilie a criação de documentos PETIC e sirva como base de dados para o desenvolvimento de outros sistemas como: aplicativos móveis, sistemas de apoio a decisão, classificação taxonômica e futuras melhorias do próprio framework.

Enquanto na qualidade de instituição de ensino, o DCOMP ministra aulas de gerenciamento de projetos, engenharia de *software* e desenvolvimento de sistemas. Todavia encontrou-se um ambiente que não se utiliza desses conhecimentos de maneira sinérgica no desenvolvimento próprio de *software*.

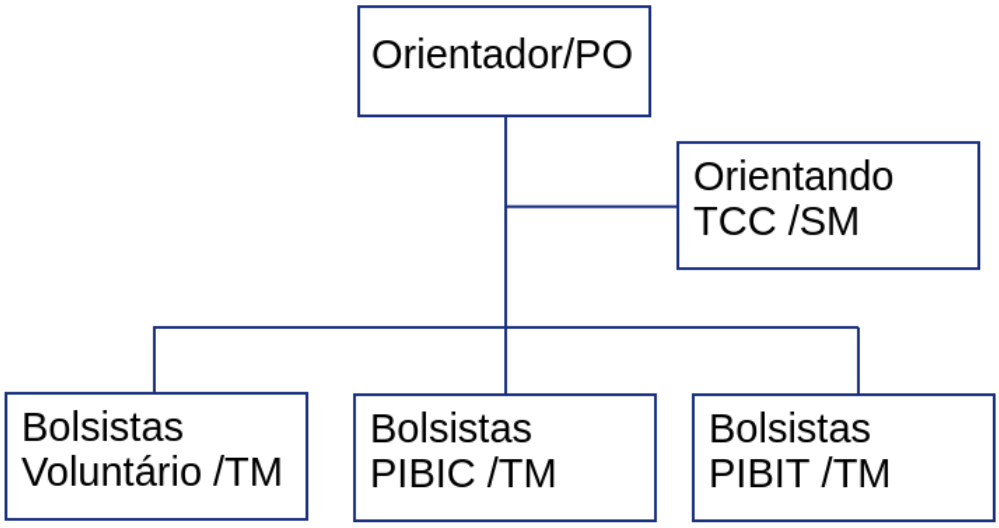
Os processos de desenvolvimento de *software* vistos definem que é necessário um planejamento prévio (mesmo que mínimo) à execução das atividades de construção do *software*. A aquisição de pessoal responsável pelo levantamento de requerimentos, comcomitante ao da equipe responsável por programar os mesmos implica que os últimos ficarão de alguma maneira ociosos, até que as primeiras definições de requerimentos sejam entregues.A natureza anualizada dos editais de projetos que financiam bolsas de pesquisa e inovação tecnológicas não contemplam essa limitação.

O projeto web-petic contou desde 2014 com pelo menos 6 alunos-pesquisadores ligados diretamente ao mesmo. Sua estrutura contempla: 1 professor-orientador, responsável por guiar os trabalhos, 2 alunos em Trabalho de conclusão do curso, 3 alunos bolsistas (remunerados e voluntários). Apesar de todos os alunos estarem diretamente ligados ao departamento o trabalho realizado por estes é feito majoritariamente de maneira remota, ou quando nas dependências da universidade de maneira autônoma e sem interação constante com outros alunos. Caracterizando assim um time virtual. Um organograma da relação entre alunos e orientador pode ser visto na Figura 6.

Eventualmente foram realizadas reuniões presenciais de alinhamento para verificar o andamento dos trabalhos. Eis uma breve linha do tempo das atividades.

Entre o início de abril de 2015 e novembro de 2015: Levantamento formal de requisitos. Todas as funcionalidades desejadas pelo sistema foram documentadas. A equipe de desenvolvimento iniciou estudo de viabilidades para definir qual a linguagem de programação a ser utilizada e qual a melhor solução de hospedagem em nuvem para a ferramenta. Esse período foi afetado pela greve das instituições federais de ensino. Entre dezembro de 2015 e março de 2016 o desenvolvimento foi iniciado de fato. No segundo período a equipe se limitou a 2 alunos bolsistas e 1 aluno de TCC. Quando foi iniciado houve a revalidação dos requerimentos desenvolvidos, configuração de ambientes, definição de interfaces de usuário. Já no início dessa

Figura 6 – Organograma PETIC



Fonte: Produzido pelo Autor

fase notou-se a necessidade de mudanças nos requerimentos.

Por fim, devido à inexistência de um processo de desenvolvimento robusto, que considerasse a equipe como virtual, que estivesse apto a entregar valor e capaz de se adaptar a face de mudanças. Fez-se necessário mudar o processo.

3.1.2.1 Metodologia de acompanhamento

As atividades de treinamento da equipe de alunos foram realizadas presencialmente. Reuniões de alinhamento com o professor orientador serviram para direcionar o trabalho e expor o andamento das atividades. Foi realizado também o acompanhamento das cerimônias de abertura,retrospecitiva e encerramento das *sprints*.

3.2 Descrição da Solução

Quando o problema foi detectado na ACONE, após uma busca de metodologias que pudessem resolver o problema gerencial, o autor procurou uma consultoria externa. A solução proposta consiste em utilizar: (i)as intervenções de formação de equipes indicadas em (SHUFFLER; DIAZGRANADOS; SALAS, 2011), como as capazes de produzir resultados, atribuição de metas e definição de atribuições;(ii) definir a comunicação clara e o respeito entre menbros da equipe como parte da criação de confiança em times virtuais (DORAIRAJ; NOBLE, 2013);e (iii) adequações de 3 metodologias comumente escolhidas pelo mercado, SCRUM Para gerenciamento do projeto, Kanban para o controle de atividades e XP para execução das atividades de desenvolvimento.

Tabela 4 – Comparação dos ambientes.

	Acone	WebPetic
RH disponivel:	4 programadores, 1 Gerente de Projeto, 1 tecnico de Infraestrutura	2 alunos desenvolvedores, 1 gerente de projeto, 1 professor orientador
Ambiente virtual de desenvolvimento:	Sim	Sim
Experiencia previa com Scrum:	Não	Não
Gerenciamento	Ineficiente, baseado em cascata.	Praticamente inexistente, baseado em cascata
Envolvimento do PO	Fraco	Fraco
Ambiente	Indústria	Acadêmico

Tabela 5 – Ferramentas Escolhidas

Necessidade	Ferramentas Escolhidas	Elementos de formação de equipe
Comunicação tempo real:	Whatsapp	Comunicação entre os membros
Gerenciamento das atividades	Quadro Trello	Atribuição de metas
Reuniões Diarias	Hangouts	Comunicação entre os membros , Atribuição de metas
Demais cerimônias da *sprint*	Hangouts (Acone) /presencial (WebPetic)	Comunicação entre os membros, Atribuição de metas
Historias de Usuario	Trello + Prototipos de tela do Pencil.	Identificar Expectativas e Responsabilidades
Pontuação dos cartões	PlanningPoker.com	Atribuição de metas

3.3 Ferramentas , Sinergias e comparações dos ambientes

Na Tabela 4 podemos ver por como os ambientes se compararam. As semelhanças entre os projetos faz do WebPetic candidato para aplicação do processo da ACONE.

Na Tabela 5 vemos que ferramentas foram escolhidas para dar suporte ao processo e como elas se relacionam com o elementos de formação de equipe.

3.4 Fases do Projeto

Este projeto está dividido nas seguintes fases:

- Atividade 0: Implantação do processo na ACONE;
  - Treinamento em SCRUM;
  - Implantação da proposta;

  - Acompanhamento dos dados;
  - Formação de equipes;
  - Metricas;

- Atividade A - Revisão da bibliografia;
  - Processos de *software*;
  - Gerenciamento de projetos;
  - Formação de equipes;
  - Metricas;

- Atividade B- Preparação do projeto WebPETIC;
  - Revisão dos requisitos;
  - Definição de UI;
  - Reunião e treinamento da equipe;

- Atividade C - Acompanhamento das atividades:
  - Cerimônias do Scrum;
  - Burdown Charts;
  - Registro de dados;

- Atividade D- Conclusão das atividades ;
  - Consolidação dos dados do projeto;
  - Treinamento em *Scrum* da próxima equipe;

4
Resultados

4.1 ACONE

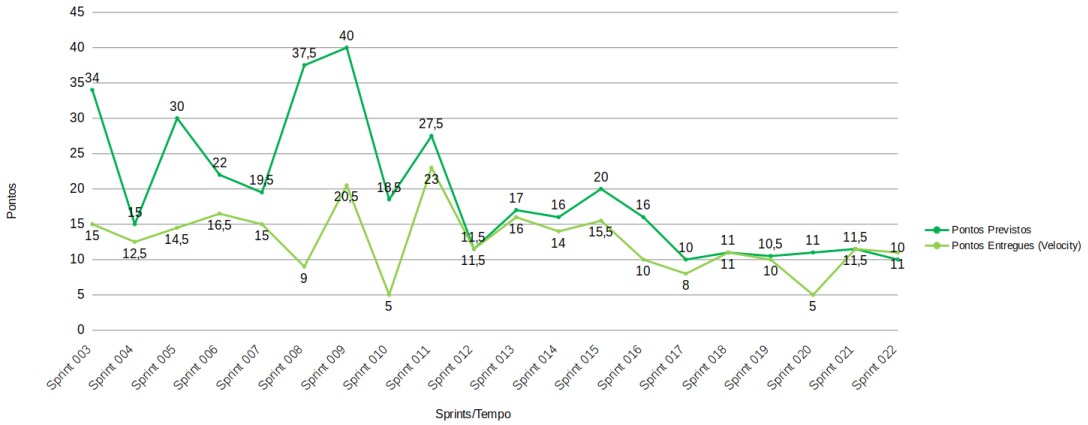
O processo na Acone mudou consideravelmente ao longo do tempo em que este trabalho foi realizado. As mudanças foram frutos de um processo de retrospectiva da sprint devidamente observado.

Uma vez identificado que o processo antigo não servia e decididos a utilizar de metodologias ágeis a equipe passou por uma preparação previa: Antes do início das *sprints* o SM promoveu uma reunião com a equipe para explicar os fundamentos do SCRUM e das metodologias que seriam seguidas. Foi explicado a definição de *sprint*, *Backlog*, Pontuação de histórias de usuário, Reunião diária, as responsabilidades e atribuições de cada papel Scrum. Foram decidido por com uma *sprint* de 1 semana, o papel de PO e SM seriam exercidos pelo autor, reuniões diárias às 18:30 via Google *Hangouts* e a pontuação de cartões seria iniciada em outras *sprints*.

O evolução da metrica de *velocity* mostra como o time evoluiu durante a aplicação do processo proposto como visto na Figura 7. Os dados completos encontram-se no apêndice A. Em seguida é discutido os eventos relevantes nas *sprints*.

Sprint 01: Contrário às expectativas do trabalho a não houve resistência da equipe à reunião diária, a equipe compareceu majoritariamente a todas as reuniões. Durante a retrospectiva da sprint os membros da equipe relataram aprovação à medida por facilitar o alinhamento entre os membros além da disciplina pessoal de alguns. A maior resistência se deu quanto a duração da sprint. Como a equipe trabalha remotamente alguns TMs elegem trabalhar durante os finais de semana quando precisavam fazer algo pessoal durante os dias da semana. Essa flexibilidade de horários é um diferencial de contração além de estar alinhado com os valores da empresa de incentivar a responsabilidade individual. Com a sprint de uma semana essa flexibilidade foi limitada, qualquer imprevisto ou interrupção poderia comprometer a meta da sprint. O PO

Figura 7 – Velocity da equipe na empresa ACONE



Fonte: Produzido pelo Autor

também teve dificuldades para praticar o refinamento de requisitos, tanto para se reunir com os clientes da empresa como para adequar Histórias de usuário que coubessem dentro de uma única *sprint*.

Sprint 02: Observando os comentários da primeira retrospectiva adotamos a sprint de duas semanas e mantivemos a reunião diária. Decidimos incluir a cerimônia de Revisão da sprint com apresentação no escritório físico da ACONE para os sócios proprietários. Durante a retrospectiva da sprint foi decidido manter a sprint de duas semanas e iniciar a pontuação dos cartões de usuário, foi arbitrado uma CRUD simples como a métrica de 1 ponto.

Sprint 03 Nessa sprint foi iniciado o acompanhamento de métricas objetivas de produtividade. Durante a retrospectiva foram debatidos comportamentos indesejados detectados na equipe. A declaração de responsável pelas tarefas logo durante a fase de planejamento de atividades, deveria ser evitada, a equipe assumiria a responsabilidade por cada cartão conforme as atividades fossem completadas. O objetivo dessa alteração foi auxiliar o time a entender quais suas metas e responsabilidades como um todo. Também foi levantado o conflito de interesses no papel de PO e desenvolvedor exercido pelo autor. O PO deve ser responsável por negociar o backlog da sprint e cobrar quando a meta da sprint não é entregue. Assim esse acúmulo de papéis em uma mesma pessoa começou a se mostrar conflitante, foi solicitado junto ao superior direto do autor que o mesmo fosse liberado das atividades de desenvolvimento para se focar.

Sprints 04-06 Durantes essas duas *sprints* o processo sofreu poucas alterações, a retrospectiva focou-se em discutir por que de uma diferença tão grande entre a pontuação prevista e a entregue, em como os pontos definidos para cada cartão era compatível com o esforço realizado e uma alteração de escopo de 9 pontos na sprint 06, equivalente a 40% do previsto. Para dar mais autonomia à equipe o autor parou de participar das reuniões diárias. Ainda durante a sprint 05 um dos TMs se demitiu.

Sprint 07 Um novo TM entrou para a equipe e iniciou sua fase de treinamento. A

retrospectiva se focou principalmente em reforçar a definição de pronto. A equipe estava se acostumando a entregar tarefas sem considerar os padrões de qualidade e testes acordados, ao final da sprint o backlog consistia em uma coleção de tarefas "quase prontas". A equipe se comprometeu à seguir o processo com mais fidelidade. Notou-se também a necessidade de melhorar a qualidade do backlog do produto, tanto em sua priorização como no conteudo de seus requisitos. Foi decidido que o PO iria lidar diretamente com o proxy no maior cliente e buscar requisitos também em outros clientes que permitissem uma maior proximidade do usuario final.

Sprints 8-11 Como pode ser visto no gráfico de pontuação entregue x pontuação prevista a equipe passou por uma fase de dificuldades antes de entender sua própria capacidade de produção. Durante essas *sprints* foi reforçado o papel da equipe na negociação da meta da sprint. A ideia de que a meta era assumida e concordada pela equipe retirou qualquer possibilidade de se eximir da responsabilidade de atingi-lá. Nesta fase os aspectos de formação de equipe se mostraram particularmente presente. A relação de respeito interpessoal foi reforçada quando a equipe notou que era tratada como um grupo de profissionais sérios, capaz de trabalhar de maneira ética e sem necessidade de microgerenciamento. O stress relacionado à não saber como seriam cobrados ou suas metas de produtividades foi substituído pela sensação de serem os responsáveis pelo seu crescimento profissional. Durante as retrospectivas notou-se que também que a pontuação entregue variou consideravelmente, entre 5 e 23 pontos uma variação 460%. Ao tentar identificar o motivo de tal variação chegamos a conclusão que eram impactos de tarefas "quase prontas" que tinha grande parte do seu trabalho realizado em uma sprint, mas eram entregues na sprint subsequente. Passamos a repontuar as tarefas pendentes de acordo com o trabalho necessario para conclusão da mesma. Particularmente na *sprint* 10 diante de um resultado particularmente baixo, consideramos abandonar o processo de SCRUM e retornar à um modelo cascata. A razão para a persistência se deu quando comparamos nossa evolução com a de outras empresa e notamos que essa fase de adequação era comum.

Sprint 12 A meta é 100% atingida pela primeira vez desde de o início do projeto. Durante a retrospectiva identificou-se que a equipe tinha melhorado sua capacidade de estimar tarefas.Esse fato foi atribuuido tanto à habituação como com a referencia de esforço de tarefas já realizadas. Uma ideia levantada como possivel melhoria do processo era utilizar desenvolvimento direcionado a testes, TDD. Definiu-se que cada historia de usuario seria complementada com criterio de aceitação e testes de aceitação.

Sprints 13-16 Apesar da diferença entre pontos previstos e entregues se manter baixa a métrica de % entregue volta a regredir. Durante a retrospectiva foi debatido a necessidade de entender que a melhoria deve ser contínua, evitando a zona de conforto que estava sendo estabelecida. Quando perguntados na retrospectiva da sprint 14 se a equipe se incomodava em ver tarefas não entregues no backlog da sprint um dos TMs respondeu que "Não, (pois) estava acostumado na faculdade a entregar 80% dos trabalhos". Na ocasião a atribuição de metas foi revisada para procurar por algum ponto de melhoria, porém nada foi detectado. Após duas *sprints*

em que a equipe foi cobrada por estar caindo a produtividade o mesmo TM da resposta citada se demitiu, o que foi visto de maneira positiva pelos demais TMs. A aplicação de TDD reduziu consideravelmente o retrabalho.

Sprint 17 Durante essa sprint contratamos um novo desenvolvedor para para entrar na equipe. O processo inicial de seleção foi feito pelo PO mas a equipe foi responsável pela palavra final da contratação. Essa experiência reforçou a capacidade auto-gerencial da equipe além da definição de novas atribuições.

Sprints 18-21 A equipe finalmente converge para uma velocity de 10.8 pontos por sprint, considerando a média das últimas 9 *sprints*. Durantes as primeiras retrospectivas discuti-se a necessidade de melhorar os ativos de conhecimento da equipe, o que foi convertido em tarefas de estudo e pesquisa de ferramentas de desenvolvimento. O Desempenho abaixo da média na sprint 20 ocorreu por uma quantidade incomum de interrupções.

Sprint 22 Desafiando a lei de Parkinson, segundo a qual o trabalho se expande de modo a preencher o tempo disponível para a sua realização, a equipe consegue entregar uma tarefa além das previamentes acordadas atingindo 110% da meta da sprint. Durante a retrospectiva notou-se que processos administrativos da empresa alheios ao desenvolvimento (Vendas, Suporte e Treinamento) começaram a limitar a produtividade da equipe. Assim a equipe resolveu por assumir mais tarefas de design e testes que antes cabiam ao PO para que o mesmo pudesse dedicar parte do tempo para ajudar a empresa a organizar seus processos.

Ao final a empresa saiu de, nas palavras do diretor, "vocês não estão atingindo nenhum prazo"para "vocês estão programando tão rápido que ninguém mais (dos setores administrativos) acompanha a evolução do sitema".

4.2 WebPetic

Os resultados no projeto WebPetic não foram tão promissores como na ACONE. Após a definição de mudar o processo de desenvolvimento foi iniciado um treinamento com os alunos nas metodologias ágeis. Durante uma reunião inicial o autor assumiu o papel de SM. Foi definido o tempo de sprint como de duas semanas, e um horário de reunião diária via hangouts que variava conforme o dia da semana por conta das disciplinas de cada discente. As tarefas do backlog seriam distribuídas para os discentes inicialmente sem estimativas de esforço e de acordo com as atividades definidas nas bolsas de cada um. Ao final dessa primeira sprint ficou claro dois problemas no modelo: (i) as atividades não poderiam ser facilmente intercambiadas, existia um alinhamento bastante rígido entre o projeto da bolsa de pesquisa e o aluno responsável pela mesma. (ii) A reunião diária dificilmente contou com todos a presença de todos os discentes. Quanto ao papel de PO este ficou sem um representante que verdadeiramente estivesse disponível. O professor responsável pelas pesquisas tinha responsabilidades com mais de vinte orientações, aulas e coordenação do mestrado, quadro devido a escassez de pessoal no DCOMP. O autor

trabalhando integralmente na empresa ACONE não dispôs de tempo para assumir também o papel de PO no projeto. Foram realizadas 3 tentativas de implantar o processo sugerido, porém, sem poder resolver os problemas de disponibilidade do PO e tarefas impessoais as tentativas foram igualmente ineficazes.

5
Conclusão

5.1 Contribuições

Este trabalho de conclusão de curso define uma metodologia de desenvolvimento de software e compara sua aplicação entre indústria e academia. A documentação das atividades e resultados obtidos deve servir como referência para à comunidade acadêmica bem como para outras empresas.

Foram identificados problemas impeditivos no âmbito de projetos de bolsa de pesquisa da UFS ao desenvolvimento de software àgil. Bem como sugestões de onde o processo desenvolvimento de *software* pode ser melhorado no DCOMP.

No âmbito acadêmico temos uma pequena contribuição secundária, não planejada nos objetivos iniciais, da compilação de um pequeno referencial teórico sobre formação de equipes e metodologias ágeis

5.2 Trabalhos Futuros

Trabalhos futuros sobre essa tema podem se dar em duas frentes, A primeira de experiências de campo , a segunda em compilação de dados já publicados.

No viés de campo, existe possibilidade é acompanhar a utilização do processo descrito neste trabalho, bem como as intervenções de formação de equipe na disciplina de engenharia de software 1 & 2 ofertadas pelo DCOMP. Diferente do ambiente com alunos bolsistas, os grupos de trabalho tem como objetivo desenvolver um único software. Os alunos podem intercambiar suas tarefas e o professor da matéria pode agir como PO dentro das horas de contato estabelecidas para aula.

Ainda no escopo de disciplinas acadêmicas existe a possibilidade de ofertar disciplina

optativa à alunos mais avançados no curso para agirem como SM/PO e formadores de equipe. Utilizando do material visto neste trabalho sinergicamente supervisionado os alunos da disciplina Engenharia de Software citada. Este trabalho pode se beneficiar da experiência docente e estrutura da matéria *Software Engineering Practice* oferecida pela Australian National University (SOFTWARE...,), detalhes podem ser encontrados no Anexo A.

Na segunda frente, existe a possibilidade de repetir o meta-estudo realizado por Salas et al. (1999) e Shuffler, DiazGranados e Salas (2011), procurando evidências recentes sobre a efetividade das intervenções estudadas. Bem como existe a possibilidade de agregar este trabalho compilando, caso possível, informações anonimizadas das métricas de produtividade de outros clientes da consultoria GoTeam, potencialmente comprovando a efetividade das intervenções de formação de equipe e metodologias ágeis.

Referências

BECK, K. et al. Manifesto for agile software development. 2001. Citado na página 23.

BECKMAN, K. et al. Collaborations: closing the industry-academia gap. *IEEE software*, IEEE, v. 14, n. 6, p. 49–57, 1997. Citado na página 12.

BEER, M. Handbook of industrial and organizational psychology. 1976. Citado na página 15.

BOEHM, B. W. Understanding and controlling software costs. *Journal of Parametrics*, Taylor & Francis, v. 8, n. 1, p. 32–68, 1988. Citado na página 19.

BRISKI, K. A. Minimizing code defects to improve software quality and lower development costs. 2008. Citado na página 20.

BULLER, P. F. The team building-task performance relation: Some conceptual and methodological refinements. *Group & Organization Management*, Sage Publications, v. 11, n. 3, p. 147–168, 1986. Citado na página 15.

DORAIRAJ, S.; NOBLE, J. Agile software development with distributed teams: Agility, distribution and trust. In: IEEE. *Agile Conference (AGILE), 2013*. [S.l.], 2013. p. 1–10. Citado 2 vezes nas páginas 16 e 32.

ECONOMIST. *The digital degree*. 2016. <http://www.economist.com/news/briefing/21605899-staid-higher-education-business-about-experience-welcome-earthquake-digital>. Citado na página 12.

HAGEN, R. Team building. *Manage ,First Quarter*, p. 26–28, 1985. Citado na página 16.

HARTMANN, D. Appropriate agile measurement: using metrics and diagnostics to deliver business value. In: IEEE. *Agile Conference, 2006*. [S.l.], 2006. p. 6–pp. Citado na página 23.

HASTIE, S. W. S. *Standish Group 2015 Chaos Report*. 2015. <https://www.infoq.com/articles/standish-chaos-2015>. Citado na página 24.

HIGHSMITH, J.; COCKBURN, A. Agile software development: The business of innovation. *Computer*, IEEE, v. 34, n. 9, p. 120–127, 2001. Citado na página 15.

HOLTON, J. A. Building trust and collaboration in a virtual team. *Team performance management: an international journal*, MCB UP Ltd, v. 7, n. 3/4, p. 36–47, 2001. Citado na página 17.

JMA. *Japan Management Association Kanban Just-in Time at Toyota: Management Begins at the Workplace*. [S.l.]: CRC Press, 1986. Citado na página 27.

KATZENBACH, J. R.; SMITH, D. K. The discipline of teams. Massachusetts, US: Harvard Business School Press, 1993. Citado na página 18.

MARCHEWKA, J. T. The fbi virtual case file: A case study. *Communications of the IIMA*, International Information Management Association, v. 10, n. 2, p. 1, 2010. Citado na página 22.

MONSALVE, S. *Bridging The Gap Between Education And The Future Workforce*. 2016. <https://techcrunch.com/2015/06/18/bridging-the-gap-between-education-and-the-future-workforce/>. Citado na página 12.

NTSB. *Aircraft Accident Report 79-05*. 1979. <https://aviation-safety.net/database/record.php?id=19780925-0>. Citado na página 16.

OWEN, R. et al. Is agile project management applicable to construction? In: *Proceedings of the 14th Annual Conference of the International Group for Lean Construction*. [S.l.: s.n.], 2006. p. 51–66. Citado na página 14.

PRESSMAN, R. S. *Software engineering: a practitioner's approach*. [S.l.]: Palgrave Macmillan, 2005. Citado na página 21.

RAD, F. T. N. K. *EXIN Agile Scrum Foundation Workbook*. EXIN, 2014. Disponível em: <https://www.amazon.com/EXIN-Agile-Scrum-Foundation-Workbook-ebook/dp/B00OZTMD52/ref=sr_1_1?s=books&ie=UTF8&qid=1491329391&sr=1-1&keywords=exin+agile+scrum>. Citado na página 27.

SALAS, E. et al. The effect of team building on performance an integration. *Small group research*, Sage Publications, v. 30, n. 3, p. 309–329, 1999. Citado 3 vezes nas páginas 15, 23 e 41.

SCHARFF, C.; VERMA, R. Scrum to support mobile application development projects in a just-in-time learning context. In: ACM. *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*. [S.l.], 2010. p. 25–31. Citado na página 14.

SCRUM.ORG. *Scrum Framework*. 2016. <https://www.scrum.org/resources/scrum-framework-poster>. Citado na página 26.

SHUFFLER, M. L.; DIAZGRANADOS, D.; SALAS, E. There'sa science for that team development interventions in organizations. *Current Directions in Psychological Science*, Sage Publications, v. 20, n. 6, p. 365–372, 2011. Citado 3 vezes nas páginas 15, 32 e 41.

SOFTWARE Engineering Practice,course overview. <http://programsandcourses.anu.edu.au/2017/course/COMP4500>. [Online; acessado em 31/03/2017]. Citado na página 41.

SOMMERVILLE, I. *Software Engineering*. [S.l.: s.n.], 2010. 56–81 p. ISSN 0014-2972. ISBN 9780137035151. Citado 5 vezes nas páginas 18, 19, 21, 22 e 23.

TERRES, M. da S.; SANTOS, C. P. dos. Confianças cognitiva, afetiva e comportamental em trocas business-to-consumer. *Revista de Administração FACES Journal*, v. 9, n. 3, 2010. Citado na página 16.

TIPPETT, D.; PETERS, J. Team building and project management: How are we doing? In: . Project Management Institute, 1995. Disponível em: <https://www.pmi.org/learning/library/team-building-project-management-doing-2026>. Citado na página 16.

TUCKMAN, B. W. Developmental sequence in small groups. *Psychological bulletin*, American Psychological Association, v. 63, n. 6, p. 384, 1965. Citado na página 17.

VICK, K. *Team-Building or Torture? Court Will Decide.* 2008. <http://www.washingtonpost.com/wp-dyn/content/article/2008/04/12/AR2008041201739.html>. Citado na página 16.

WERNHAM, B. *Agile project management for government*. [S.l.]: Maitland and Strong, 2012. Citado na página 22.

YEN, D. C. et al. Differences in perception of is knowledge and skills between academia and industry: findings from taiwan. *International Journal of Information Management*, Elsevier, v. 23, n. 6, p. 507–522, 2003. Citado na página 12.

Apêndices

APÊNDICE A – Dados dos resultados na empresa ACONE

Tabela 6 – % Tarefas Entregas na empresa ACONE

Sprints / Indicadores	Qtd Tarefas Previstas	Qtd Tarefas Entregues	% Tarefas Entregues
Sprint 003 06/jul - 20/jul	18	12	67%
Sprint 004 21/jul - 03 ago	12	6	50%
Sprint 005 04/ago - 17/ago	29	15	52%
Sprint 006 18/ago - 31/ago	22	15	68%
Sprint 007 01/set - 15/set	18	9	50%
Sprint 008 15/set - 28/set	14	6	43%
Sprint 009 05/out - 19/our	19	8	42%
Sprint 010 19/out - 02/out	15	6	40%
Sprint 011 02/nov – 15/nov	18	15	83%
Sprint 012 16/nov - 30/nov	11	11	100%
Sprint 013 09/dez - 23/dez	17	16	94%
Sprint 014 04/jan – 26/jan	23	22	96%
Sprint 015 26/jan – 15/fev	17	13	76%
Sprint 016 16/fev – 29/fev	15	9	60%
Sprint 017 29/fev – 14/mar	9	8	89%
Sprint 018 14/mar – 28/mar	13	13	100%
Sprint 019 28/mar – 11/abr	10	6	60%
Sprint 020 11/abr – 25/abr	9	3	33%
Sprint 021 25/abr – 09/mai	17	18	106%
Sprint 022 09/mai – 23/mai	12	12	100%

Tabela 7 – *Velocity* na empresa ACONE

Sprints / Indicadores	Pontos Previstos	Pontos Entregues (Velocity)	% Pontos Entregues
Sprint 003 06/jul - 20/jul	34,0	15,0	44%
Sprint 004 21/jul - 03 ago	15,0	12,5	83%
Sprint 005 04/ago - 17/ago	30,0	14,5	48%
Sprint 006 18/ago - 31/ago	22,0	16,5	75%
Sprint 007 01/set - 15/set	19,5	15,0	77%
Sprint 008 15/set - 28/set	37,5	9,0	24%
Sprint 009 05/out - 19/our	40,0	20,5	51%
Sprint 010 19/out - 02/out	18,5	5,0	27%
Sprint 011 02/nov – 15/nov	27,5	23,0	84%
Sprint 012 16/nov - 30/nov	11,5	11,5	100%
Sprint 013 09/dez - 23/dez	17,0	16,0	94%
Sprint 014 04/jan – 26/jan	16,0	14,0	88%
Sprint 015 26/jan – 15/fev	20,0	15,5	78%
Sprint 016 16/fev – 29/fev	16,0	10,0	63%
Sprint 017 29/fev – 14/mar	10,0	8,0	80%
Sprint 018 14/mar – 28/mar	11,0	11,0	100%
Sprint 019 28/mar – 11/abr	10,5	10,0	95%
Sprint 020 11/abr – 25/abr	11,0	5,0	45%
Sprint 021 25/abr – 09/mai	11,5	11,5	100%
Sprint 022 09/mai – 23/mai	10,0	11,0	110%

Tabela 8 – Pontos de Interrupção na empresa ACONE

Sprints / Indicadores	Interrupções Entregues (Qtd.)	Interrupções Entregues (Pontos)
Sprint 003 06/jul - 20/jul	0	0
Sprint 004 21/jul - 03 ago	1	2
Sprint 005 04/ago - 17/ago	5	3,5
Sprint 006 18/ago - 31/ago	8	9
Sprint 007 01/set - 15/set	3	3
Sprint 008 15/set - 28/set	2	2
Sprint 009 05/out - 19/our	7	4
Sprint 010 19/out - 02/out	0	0
Sprint 011 02/nov – 15/nov	2	1.5
Sprint 012 16/nov - 30/nov	2	1,5
Sprint 013 09/dez - 23/dez	3	1.5
Sprint 014 04/jan – 26/jan	0	0
Sprint 015 26/jan – 15/fev	5	3
Sprint 016 16/fev – 29/fev	6	4
Sprint 017 29/fev – 14/mar	0	0
Sprint 018 14/mar – 28/mar	5	4
Sprint 019 28/mar – 11/abr	2	1
Sprint 020 11/abr – 25/abr	11	5,5
Sprint 021 25/abr – 09/mai	1	1
Sprint 022 09/mai – 23/mai	0	0

Figura 8 – *Velocity* da equipe na empresa ACONE

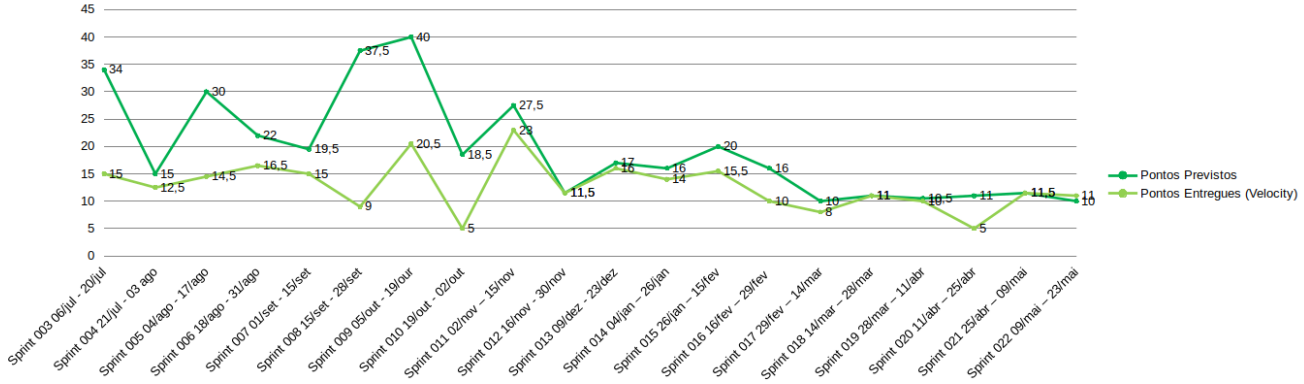


Figura 9 – Porcentagem de pontos entregue na empresa ACONE

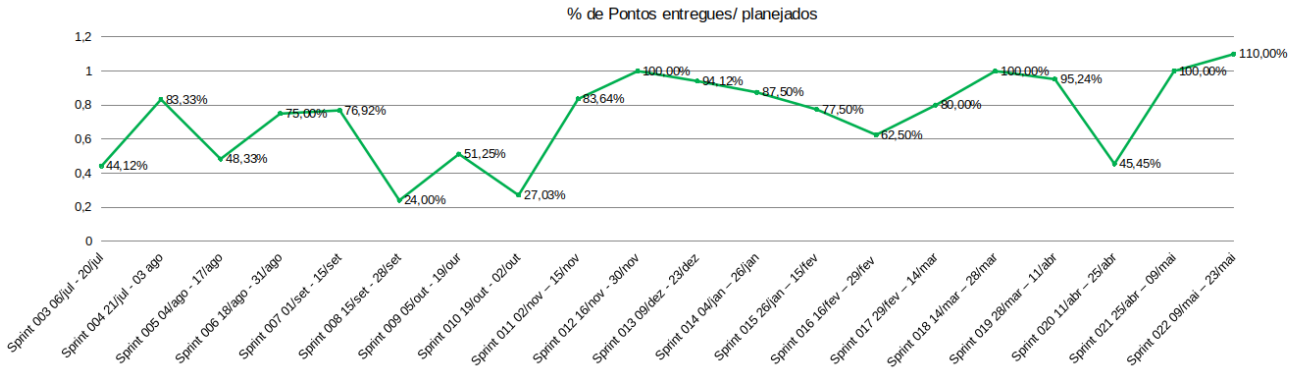
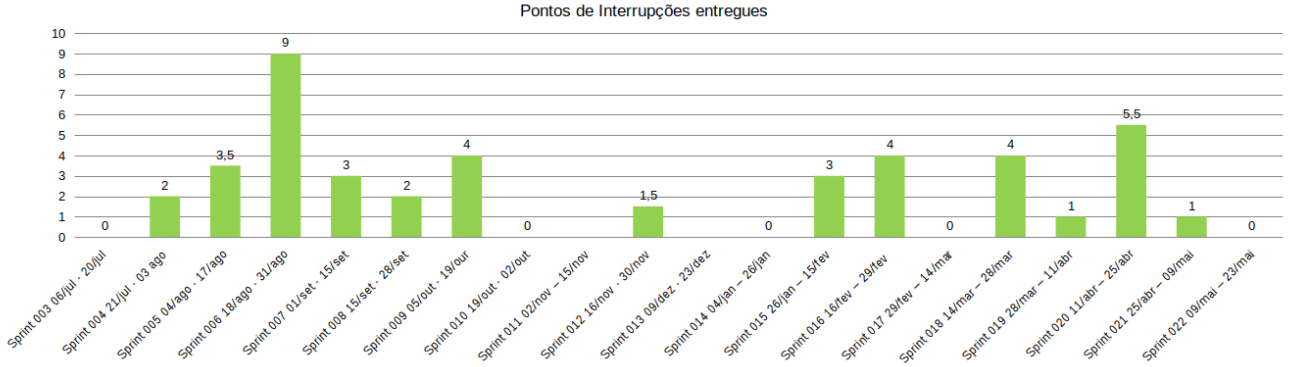


Figura 10 – Quantidade de pontos de interrupção entregues na empresa ACONE



Anexos

ANEXO A – *Software Engineering Practice,ANU COMP4500 course overview*

An undergraduate course offered by the Research School of Computer Science.

This course exposes students to professional software engineering practice and leadership through the development of a software system for an industry, government or university based customer, or by engaging with the ACT innovation community and students across the university to create a software-based start-up business. Students will lead small teams (COMP3500 students) to plan (define, estimate, schedule) and manage an appropriate set of activities to ultimately deliver a software product. The implementation part of the project will include monitoring, measuring, tracking, and managing change.

A.1 Learning Outcomes

Upon completion of this course, students will be able to:

- Work as an effective team manager of a small team of peers to implement a software based solution that delivers measurable value to an industry or university client.
- Develop life-long learning through reflection, as demonstrated through continual reflection on the software development lifecycle and team work processes experienced throughout the year.
- Exhibit an awareness of and an ability to employ- team formation strategies and stages leading to the development of high performing, self-managing teams;- sound meeting practice; and- how personality traits can impact upon team performance and how to use individual traits to achieve the most from team work.
- Make and defend sound engineering decisions.
- Communicate effectively, orally and in writing, with peers, supervisors and commercial clients/stakeholders.
- Creatively identify and implement a solution to a complex problem that exists within the domain of ICT.
- Participate effectively in project and artefact reviews with peers, supervisors and clients/stakeholders.

- Through effective implementation of appropriate activities and processes, demonstrate a sound understanding of the importance of project management, configuration and risk management processes when undertaking a software development project.
- Through appropriate choice and implementation of activities associated with each phase, demonstrate a sound understanding of the software development lifecycle (SDLC).
- Through appropriate choice and tailoring of standards, demonstrate a sound understanding of the role and importance of standards in software development.
- Co-ordinate and develop presentations, including demonstrations, to an audience of peers, clients and supervisors.

A.2 Indicative Assessment

Three group project reviews (75%), Group Poster and Showcase (10%), Individual Reflection (15%) The ANU uses Turnitin to enhance student citation and referencing techniques, and to assess assignment submissions as a component of the University's approach to managing Academic Integrity. While the use of Turnitin is not mandatory, the ANU highly recommends Turnitin is used by both teaching staff and students. For additional information regarding Turnitin please visit the ANU Online website.

A.3 Workload

Annual course. Student enrolls in Semester 1 and Semester 2. Around 50 contact hours across Team Formation Day, introductory lectures, tutor meetings, project reviews and public showcase. Around 210 hours of group project work and individual study.